RDATA Periodicity

How to access data infrequently

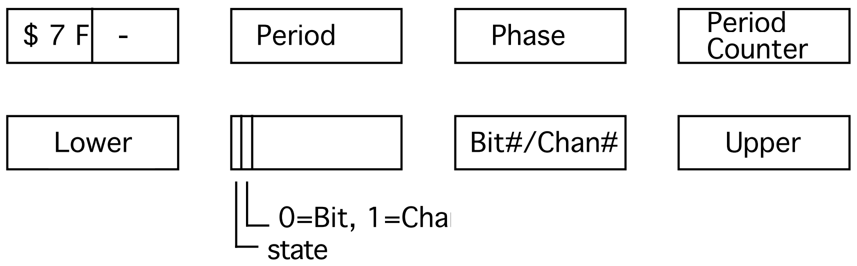
Sep 14, 1989

The Read Data Access Table (`RDATA`) has traditionally been scanned on each 15 Hz cycle to read all the data from the hardware interfaces into the system's datapool. The analog data is stored in the `ADATA` table and the binary status data is stored in the `BBYTE` table. But there can be reasons why the data should not all be read at 15 Hz. One reason may be that it takes too much time to read everything every cycle. Another reason is that the very act of reading some hardware interfaces can cause noise interference. The liquid argon temperatures in D0 are an example of the latter case. This note describes a plan for accommodating access to data at more leisurely intervals.

In order to arrange for less frequent execution of some entries in the `RDATA` table, some means of keeping a counter is required. We can assume that the table will be scanned at 15 Hz, but some entries may not be executed each time. To pick a concrete example, let's say that we want to read some hardware at 15 Hz and some other hardware at 1 Hz, but there is a *lot* of hardware that must be read at 1 Hz. We want to schedule the accessing of some of that hardware on different cycles. This means that we want to specify a "phase" value for sequencing the components of the large amount of 1 Hz data. So we need a way to specify a period, a counter to count out the period value, and a "phase" value to initialize this period counter at reset time.

Consider groups of entries in `RDATA` which should be processed with the same period and phase. Use a special entry which acts as a sort of header for the following entries and specifies this period information for the following group of entries. The group size includes all entries until the next special period-holding entry or the end of the table, whichever comes first.

Suppose we use the following period-holding format:



Note that the `Period` is specified as a 16-bit field. This allows for periodic data collection to be as infrequent as approximately every hour. The optional `Bit#` or `Chan#` and `state` value (in the sign bit of the word) can be used to condition the processing of the entire group of `RDATA` entries which follow this period entry. The `Bit#/Chan#` must be nonzero to enable this option. When using this feature, one must take care to have collected the binary status data or the analog channel reading data, on which the `Bit#/Chan#` test depends, *before* the period entry which needs to reference it.

For the `Bit#` case, the following group of `RDATA` entries is enabled when the state of the indicated Bit matches the `state` bit. For the `Chan#` case, when the `state` bit=0, the group is enabled when the reading of the indicated `Chan` is within the range of values given by the `Lower` and `Upper` words. If the `state` bit=1, the group is enabled when the reading is outside the range.

At reset time, the system scans the entries in the `RDATA` table for these period-holding entries. For each one it copies the `Phase` value into the `Period Counter`. (If the `Phase` value is larger than the `Period` value, it merely zeros the `Period Counter`.)

At 15 Hz time, when the `Update` task is reading the data from the hardware, the `RDATA` table is scanned for a period-holding entry. All entries are ignored until the first such entry. If the `Bit#/Chan#` word is nonzero, apply the indicated test above. If the test fails, set a flag to ignore the following entries until the next period entry; if it is successful, clear the `Period Counter`. Next, if the `Period Counter` is zero, copy the `Period` value into the `Period Counter` and set a flag to enable processing of the following entries. In any case, decrement the `Period Counter`. For the simple case of 15 Hz, the `Period` would be 1, and the `Phase` would be 0. Viewed on the Memory Dump page, the `Period Counter` would seem to always have the value 0. For longer periods, the `Phase` may be set to any value from 0 up to (`Period – 1`), and the `Period Counter` would exhibit its decrementing behavior.

This capability for execution of various accesses to hardware interfaces on an infrequent basis can also be used for other infrequent scheduling within the VME system. The meaning of each entry type in the `RDATA` table is fairly open-ended, and new code can be added to do other jobs.